Desarrollo FrontEnd con JavaScript

s04 – jQuery, Promises y $.Deferred

Manipulación de DOM

Manejo de eventos

jQuery

Plugins

Ajax

Manipulación de DOM

```
var container = $('.container');

container.addClass('user-container');

container.css({
  width: '120px',
  height: '80px'
});
```

Manejo de eventos

```
$('#add-container').on('click', function(e) {
  e.preventDefault();

  $('body').append('<div class="container"></div>');
});
```

Plugins

```
$.fn.addContainer = function() {
  var container = $('<div class="container"></div>');
  container.addClass('user-container');
  container.css({
    width: '120px',
    height: '80px'
  });

  return $(this).append(container);
};

$('body').addContainer();
```

Ajax con jQuery

$.ajax

```
$.ajax({
  url: '/tweets.json',
  data: {
    query: 'cappuccino'
  },
  success: function(data) {
    console.log(data.length + ' tweets');
  }
});
```

$.getJSON

```
$.getJSON('/tweets.json', { query: 'cappuccino' },
function(data) {
    console.log(data.length + ' tweets');
});
```

$.Deferred

Promises + jQuery

Promises

- Permite asignar varios callbacks a una petición asíncrona.
- Separa la petición de los callbacks de éxito y error.
- Encadena peticiones

Promises con jQuery

```
$.ajax({
  url : '/tweets.json',
  data: {
    query: 'cappuccino'
  },
  success : function(tweets) {
    alert('Tweets: ' +
tweets.length);
  }
});
```

```
var promise = $.ajax({
  url : '/tweets.json',
  data: {
    query: 'cappuccino'
  }
});


promise.done(function(tweets) {
  alert('Tweets: ' +
tweets.length);
});
```

Promises – done

```
var promise = $.getJSON('/tweets.json', { query:
'cappuccino' });

promise.done(function(tweets, status, xhr) {
  alert('Tweets: ' + tweets.length);
});

promise.done(function(tweets, status, xhr) {
  alert('First tweet: ' + tweets[0]);
});
```

Promises – fail

```
var promise = $.getJSON('/tweets.json', { query:
'cappuccino' });

promise.fail(function(xhr, status, error) {
  console.log('Error: ' + status);
});
```

Promises - then

```
var promise = $.getJSON('/tweets.json', { query:
'cappuccino' });

promise.then(function(tweets, status, xhr) {
  alert('Tweets: ' + tweets.length);
}, function(xhr, status, error) {
  console.log('Error: ' + status);
});
```

$.when

```
var ajaxTweets =
$.getJSON('http://coffeemaker.herokuapp.com/twitter.json?q=cappucci
no');
var ajaxPhotos =
$.getJSON('http://coffeemaker.herokuapp.com/instagram.json?q=latte'
);

$.when(ajaxTweets, ajaxPhotos).done(function(argsTweets,
argsPhotos) {
  var tweets = argsTweets[0];
  var photos = argsPhotos[0].data;
  console.log("Tweets", tweets.length);
  console.log("Photos", photos.length);
});
```

Deferred Object

- Crea funciones asíncronas.
- Devuelve un promise.
- `resolve`: dispara los callbacks en **done**
- `reject`: dispara los callbacks en `fail`

```
var deferred = new $.Deferred();

deferred.done(function(user) {
  Users.add(user);
});

deferred.done(function(user) {
  console.log('Added user');
});

deferred.resolve(new User());
```

$.Deferred - reject

- Con `reject` se disparan todos los callbacks definidos con `fail`.

```
var deferred = new $.Deferred();

deferred.fail(function(message,
user) {
  console.log('Error: ' +
message, user);
});

deferred.reject('Error at adding
new user', new User());
```